First Hit Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L7: Entry 1 of 1 File: JPAB Jan 23, 1998

PUB-NO: JP410020783A
DOCUMENT-IDENTIFIER: JP 10020783 A
TITLE: RANDOM NUMBER GENERATING DEVICE, RANDOM NUMBER GENERATING SYSTEM AND CIPHER COMMUNICATION SYSTEM

PUBN-DATE: January 23, 1998

INVENTOR-INFORMATION:

NAME	COUNTRY
WATANABE, EIJI	
TAKE, KOJI	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
METEOOLA SYST KK	
KK AIRU	

APPL-NO: JP08169869
APPL-DATE: June 28, 1996

INT-CL (IPC): G09 C 1/00; G09 C 1/00; G06 F 7/58; H04 L 9/10; H04 L 9/26

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a system that can materialize a one-time pad cipher on a personal computer in open network environment and can throw away a common key.

SOLUTION: The initial value Xi is inputted as a secret key (S1), and chaotic start-up (X1→x2→...→xn) is generated one after another while performing chaotic mapping operation (S2, S3). Upon the completion of this generation, the computed value at the time of completion is stored as the next initial value (S4), that is, stored as a new secret key. On the other hand, a binary random number by unidirectional compression is generated using an orbit generated in the step S2 (S5), and this binary random number is outputted with a random number stream as a cipher (a dynamic work key) (S6).

Previous Doc Next Doc Go to Doc#

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－20783

(43)公開日　平成10年(1998)１月23日

(51)Int.Cl.⁶		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０９Ｃ	1/00	６５０	7259－5Ｊ	Ｇ０９Ｃ	1/00	６５０Ｂ
		６１０	7259－5Ｊ			６１０Ｄ
Ｇ０６Ｆ	7/58			Ｇ０６Ｆ	7/58	Ａ
Ｈ０４Ｌ	9/10			Ｈ０４Ｌ	9/00	６２１Ａ
	9/26					６５９

審査請求　未請求　請求項の数９　ＯＬ　（全 14 頁）

(21)出願番号　特願平8－169869

(22)出願日　平成８年(1996)６月28日

(71)出願人　591223231
メテオーラ・システム株式会社
神奈川県横浜市港北区高田町1549番地

(71)出願人　596059163
株式会社アイル
東京都中野区本町２丁目50番８号

(72)発明者　渡邊　榮治
神奈川県横浜市港北区高田町1549番地

(72)発明者　武　弘司
東京都小金井市貫井南町４丁目４番８号

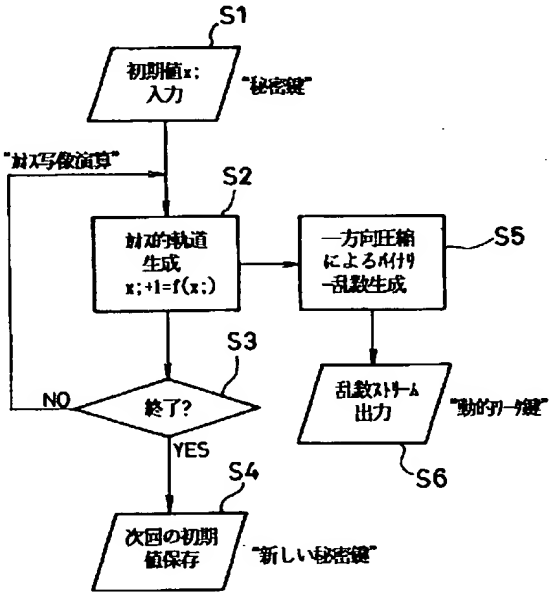
(74)代理人　弁理士　三好　秀和　（外８名）

(54)【発明の名称】　乱数生成装置及び乱数生成システム並びに暗号通信方式

(57)【要約】　（修正有）

【課題】　オープンなネットワーク環境下にあるパソコン上でＯｎｅ－Ｔｉｍｅ　Ｐａｄ暗号を成立させること、共通鍵を使い捨てできる方式を得ること。

【解決手段】　初期値Ｘｉを「秘密鍵」として入力して（Ｓ１）、カオス的写像演算を行いながらカオス的起動（$X_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$）を次々と生成する（Ｓ２、Ｓ３）。そして、この生成が終了すると、その終了時点での演算値を次回の初期値として保存する（Ｓ４）。つまり、新しい秘密鍵として保存する。一方、ステップＳ２で生成された軌道を用いて一方向圧縮によるバイナリー乱数を生成し（Ｓ５）、これを乱数ストリームを暗号（動的ワーク鍵）として出力する（Ｓ６）。



1

【特許請求の範囲】

【請求項1】　乱数を生成する目的で、コンピュータに通常備わっている2進表現レジスターとは異なり、10進数を記憶するレジスターでありながら、その一桁がさらに任意の桁の数値を収容できるレジスターとなっている「入れ籠構造」を成した特殊レジスターを備え、

該特殊レジスターの個数と桁数とを任意に調整できる有限精度空間（Special Finite Precision；SFPと略記する）を備え、

該SFPのレジスターが表現する任意の10進数の数値から0／1のバイナリーコードを作り出す一方向変換メカニズムを備え（一方向の圧縮関数と同義）、

該SFPのレジスターからSFP自身のレジスターへの写像を実行する非線形関数を備え（カオス写像と略記する）、

上記カオス写像をくり返して（カオス写像演算と言う）該レジスターの数値を$X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_n$と次々と生成し（この数列をカオス的軌道と略記する）、

これに伴なって前記0／1のバイナリーコードのストリームを作り出すことにより、

上記カオス的軌道の任意の軌道値（通常、初期値）を「秘密鍵」として利用する一方、前記バイナリーコードのストリームを暗号・復号化する「ワーク鍵」として暗号通信に利用することを可能としたことを特徴とする乱数生成装置。

【請求項2】　乱数を作る目的で、相当数桁の10進数を記憶できる10進レジスターを備えることを特徴とし、

該10進数値の、非線形関数によるそれ自身への写像をくり返すカオス写像演算を備え、

該カオス写像演算の結果生み出されるカオス的軌道の任意の軌道値を「秘密鍵」とする一方、

該秘密鍵からさらに一方向圧縮関数を介して作り出したビットストリームを、「ワーク鍵」とする、

暗号通信を行う乱数生成システム。

【請求項3】　ネットワーク上の端末群が前記1記載の乱数生成装置又は2記載の乱数生成システムを備えて、一方、前記秘密鍵を保管したICカードなどの携帯媒体とそれを脱着自在とする入出力装置を備え、あるいは前記端末の記憶装置に該秘密鍵とそのユーザーIDのテーブルから成る鍵ファイルを備え、

前記乱数生成システムの秘密鍵を上記特定の端末群が「共有する」ことによって該端末間に暗号通信を可能ならしめたことを特徴とする暗号通信方式。

【請求項4】　前記暗号通信方式で共有されている秘密鍵すなわち共通鍵［X_1］は、当該通信の作法とタイミングを規定した暗号通信プロトコルの中に組み込まれて、

その「生成」、「利用」、「更新」の時を持つことを特徴とした請求項3記載の暗号通信方式。

2

【請求項5】　前記請求項4の暗号通信方式にて、

共通鍵［X_1］が当該暗号通信プロトコルによって「更新」される場合、

当該暗号通信プロトコルは、共通鍵［X_1］のカオス的軌道［X_1、X_2 …X_i］に含まれる軌道値［X_{i+1}］を、新しい秘密鍵として指定し、該秘密鍵をもって共通鍵の更新を行なうことを特徴とした請求項3又は4記載の暗号通信方式。

【請求項6】　前記暗号通信方式にて、前記共通鍵の更新を、少なくとも二つの端末が同期して実施するために、

該通信プロトコルは、通信状態のアプリケーションソフトを「送信モード」か「受信モード」のどちらかの状態に置く一方、

そこで送受信される情報は、固定長部と可変長部とから成る統一された通信文フォーマットに載せられねばならず、その固定長部には「今回使用する共通鍵［X_1］」と「次回使用予定の共通鍵［X_{i+1}］」と「付加情報」の三つの情報ユニットが載り、対応した可変長部には「本文」が載ることを特徴とした請求項3、4又は5記載の暗号通信方式。

【請求項7】　前記暗号通信方式にて、

該通信プロトコルの送信モードでは、固定長部の「今回使用する共通鍵［X_1］」を用いて、該共通鍵自体［X_1］と次回使用予定の共通鍵［X_{i+1}］及び付加情報の三つの情報ユニットを一緒に暗号化することを特徴とした請求項3、4、5又は6記載の暗号通信方式。

【請求項8】　前記暗号通信方式にて、

受信モードの下で暗号文を受け取った端末は、自端末で保管していた共通鍵［X_1］.decによって（受信モード側の鍵を［X］.decと記号する）暗号文を復号し、

復号された方の共通鍵のデータ［X_1］.encと（送信モード下の鍵を［X］.encと記号する）受信モード下の共通鍵［X_1］.decとを比較し、

両者が等しければ該受信モードの端末は送信モードの端末を正当な暗号通信の契約者と認定する一方、

両者が等しくなければ、受信モードの端末は送信者を「成りすまし」、「詐称」の可能性があるものとして例外処理を行なうか又は、送信者が「不正な通信歴」を経過した可能性があるとして例外処理を行なうなどして、

該共通鍵を通信プロトコルに組み入れたことによって送信者の「認証」を可能としたことを特徴とする請求項3、4、5、6又は7記載の暗号通信方式。

【請求項9】　前記6、7、8の請求項に該当する暗号通信方式にて、

通信文を暗号化ファイルに保管する場合、固定長部の三つの情報ユニットを共通鍵で暗号化するのではなく、

固定長部の付加情報部に本文を暗号化する際に使った秘密鍵を置き、該固定長部全体の暗号化をICカード等に保管されている別途秘密鍵（マスター鍵）によって実施

することを特徴とした請求項6、7又は8記載の暗号通信方式。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】オープン環境下のネットワーク、特にインターネットとか衛星通信などに於ける暗号技術、鍵配送技術、認証技術、暗号通信プロトコルなどに関する。

【0002】

【従来の技術】一般に近代的な暗号技術では、暗号・復号をするときに「鍵」を用いる。この「鍵」管理の信頼性と扱い易さを両立させることがインターネット等における様々な問題解決に役立つ。

【0003】この「鍵」管理の信頼性を高めるために、「鍵」の階層を作って「鍵」の管理を行なうことが知られている。例えば、「暗号論入門　共立出版株式会社、岡本栄司　1994年10月1日」（以下文献1という）によれば、暗号鍵は図9に示すように階層化されている。1番下の鍵をデータ暗号鍵あるいはワーク鍵といい、下から2番目以上の鍵を鍵暗号化鍵、最上位の鍵をマスター鍵という。ワーク鍵で実際のデータの暗号化／復号化を行う。下からn－1段目の鍵を下からn段目の鍵暗号化鍵で暗号化して相手側に送る。

【0004】図9におけるマスター鍵はオフライン等の手段により暗号化側と復号化側で伝えあう必要がある。階層数が1のときは、ワーク鍵自体をオフラインなどで伝えあうことになる。

【0005】階層数を2として、対称暗号をデータ暗号に用い、そのときのワーク鍵を非対称暗号で暗号化して相手側に伝えることはよく行われている。このハイブリッドタイプのメリットは、高速処理可能な対称暗号で大量のデータを暗号化できること、およびそのときのワーク鍵を非対称暗号で暗号化すれば、マスター鍵を公開鍵とすることができることである。全体としてみると、暗号処理速度が大きくかつ暗号強度が高い暗号系を構築できる。

【0006】この説明を見ても判るように、「鍵」の扱い易さと信頼性とは、トレードオフしている。対称暗号は、特に共通鍵を用いるので扱い易さは合格であるが、同じ鍵を何度も用いるので、信頼性はそれだけ低下せざるを得ない。特に、バリューデータ（電子マネーなど）のような高度なセキュリティーには不向きとされている。

【0007】この欠点を克服している対称暗号が古くから知られている；One－TimePad暗号とかVernum暗号と言われている技術である。

【0008】これは、図10に示すような手順で平文を1ビットずつ乱数ストリームのビット・ストリームと排他的論理和（XOR）するという、ごく簡単な手続きの技術であり、一般的に「ストリーム暗号」（stream cipher）と言われている。

【0009】このような技術においては、特に乱数列が真正乱数であるとき「暗号学的にもっとも安全である」と考えられている。

【0010】また、上記のOne－Time Padとは、「乱数列を使い捨て」とするのであるが、これは、共通鍵の考え方に直すと、1回の通信毎に「共通鍵を捨てる」ということを意味する。

【0011】このOne－Time Pad暗号は、しかし、民間では利用できない。その理由は、真正乱数をソフトウェアで作れないことと、乱数を使い捨てにするということで極めて扱いが困難になるからである；最高の秘密通信にのみ使用されているということである。

尚、真正乱数とは、0と1とが等確率に出現するビット列を言う。

【0012】このOne－Time Pad暗号を民間でも利用できるようにしようと考えると、

① ソフトウェアで真正乱数を作れること（少なくもループしない乱数ストリームを作れること）。

② 乱数ストリームを鍵によって100%再現できるようにすること。

③ 差別化された鍵の個数すなわち鍵空間は2^{128}個以上のサイズでなければならない。鍵空間が2^{128}個以上のサイズというのは、「UNIX MAGAZINE1995年10月号　P70」（以下文献［3］という）によれば、一秒間当たり10^9個の命令を実行できる計算機を100億の人間がそれぞれ持ち、それぞれいっせいに計算を始めたとすると、8000億年もかかることを述べている。

④ 鍵は、何らかの方法で使い捨てにする。何らかの方法とは、ハイブリッド・タイプではない、扱い易い新しいスキームである。

【0013】以上少なくも①～④を満たす新しい乱数ストリーム生成ソフトウェアを必要とする。ところが、従来の乱数生成ソフトウェアは、乱数を作るのに“乱数”すなわち真正乱数を入力するか、デタラメなタイプ入力をさせて“偶然性”を人が作り出してやらねばならないという“にせ者”である。実は、対称暗号の技術は、こういう疑似乱数生成法を要素技術として使っているので、安全とは考えられていない。

【0014】従って、使い難いけれども、安全性を採る場合は物理乱数を加工して真正乱数を作り出しているのである。

【0015】この物理乱数とは、自然現象から抽出した乱数であり、真正乱数とはならないが、それに近い信頼性があるとされている。

【0016】

【発明が解決しようとする課題】このように、オープンなネットワーク環境下にあるパソコン上で（1）真正乱数を無尽蔵に生成すること、（2）共通鍵を使い捨てに

5

すること、という二つの要請に答えることは従来技術ではできない。従って、One－Time pad暗号はコンピュータ技術の問題外であった。

【0017】そこで前項に述べた①～④の要請に応える乱数生成ソフトウェアの考案と、共通鍵を使い捨てにする新しい暗号通信方式を考案することで、オープンなネットワーク環境下にあるパソコン上でも（A）One－Time Pad暗号を成立させること、（B）共通鍵を使い捨てにすること、という二つの要請に答える。また、上記（1）と（2）をベースにして、（3）通信当事者の認証技術の問題及び、（4）イントラネットの普及とともに、組織内の情報流出を防ぐ方策がないという問題；現在外部からのアクセス制限を行なうことはファイアーウォールでしているが、内部からの流出は防げない。これらの（1）～（4）の問題を解決する。

【0018】

【課題を解決するための手段】請求項1と2は、乱数生成ソフトウェアの構成法に関し、従来に無い新しいスキームに関する、40年以上前に考案され、de facto standardになったLehmer generator以来、乱数生成には2進表現のレジスターが用いられてきた。これはコンピュータの限界に挑んできたという明しでもある。しかし、当考案では、このコンピュータの限界を楽に越える手段として、「10進数を記憶するレジスターでありながら、その一桁がさらに任意の桁の数値を収容できるレジスターとなっている“入れ籠構造”を成したレジスター」を考案した。この考案が乱数を作る上でいかに画期的であるかを以下に説明しなければならない。

【0019】カオス的現象についてはコンピュータの助けによって解明されてきたが、カオスそのものをコンピュータの中で実現することには、未だ成功していない。その根本的な理由は、カオスの舞台となっているところは、「連続量」の世界であるからである。；区間［0，1］の間に埋っている実数の世界である。連続量は図12に示すようにさらに二つのカテゴリーに分けられる；図12の①は整数のグループを無限に並べた世界を想像すればよい。図12の②は、グループの中味が数えられないというグループが無限にあるということで、さらに想像不可能な世界である。

【0020】一方、コンピュータの世界は、レジスターの能力で決まる整数のグループが一つ有るだけである；これを有限精度空間と言う。従って、カオスはもちろん生れないし、カオス的現象も限られた範囲でしか作れない。こういう条件で現代の乱数生成法は、コンピュータのレジスターという2進表現の「有限な非連続量」の世界の中で、その限界に挑んできた技術であると言える。

【0021】これに対して当発明の原点は、コンピュータの有限精度を一つの「構造」と考えて、この構造自身の「自己相似形」をその構造の中に組み込むという画期的メカニズムを導入した。これによって連続量、特に「可算個の無限」をコンピュータの中で取り扱うことが可能になった。「連続量を非連続量で表現する」ということを具体的に述べると、パソコンのレジスターで扱える整数値は、せいぜい

$10^9 < 2^{31} \sim 2^{32} < 10^{10}$

である。

【0022】今、0から10^9の数のグループを一つの有限精度空間と受け入れて、この非連続量によって可算個の無限、10^L（$L \to \infty$）を法とする線分［0，1］を表現してみよう；

①$10^L$なる数値を10^9個のグループに分ける、

$gr1 = 10^L / 10^9 \Leftrightarrow \Delta = 1/10^9$

但し、Δは線分を10^9個の微小線分Δに分けたものである。

②gr1をさらに10^9個のグループに分ける。

$gr2 = 10^L / 10^9 \times 10^9 \Leftrightarrow \Delta^2 = 1/10^9 \times 10^9$

以下同様にして、次々と10^9個のグループを作る；

③　$grn = 10^L / 10^{9n} \Leftrightarrow (\Delta)^n = 1/10^{9n}$

このようにして、可算個の無限の中に「構造」を入れる。

【0023】このネスティング（入れ籠）の回数nを増やすことによって、小さな数10^9によっていくらでも大きな数10^L（$L \to \infty$）を取り扱えるようになる。あるいは［0，1］区間の写像であれば、いくらでも小さな数$(\Delta)^n \to 0$を取り扱えることとなるのである（これは非連続量の中のフラクタル構造という側面から、カオス発生のメカニズムを論じるべきであるが、他に譲る）。

【0024】従って、請求項1の冒頭の「入れ籠構造を成したレジスターを備え、該特殊一様レジスターの個数と桁数とを任意に調整できる有限精度空間を備え、」という手段は、カオス発生の舞台を用意するという（ネスティングn→∞；実用上n＝several）画期的な発明に関するのであることか判る。これ以降の手段は、乱数をバイナリーコードで出力し、かつ暗号学上安全であることを保証する「一方向の圧縮関数」を備え、及びカオス写像を実行する非線形関数を備え、かくしてカオス的軌道（通常ストレンジ・アトラクターという）を生成して、この初期値をもって、カオス的軌道と乱数列とを制御するという手段と成す。

【0025】すなわち、上記カオス的軌道値の任意の軌道値、通常初期値を「秘密鍵（private key）」又は「マスター鍵」として利用するなら、一方向圧線関数の出力であるバイナリーコードのストリームを制御して暗号・信号を直接実行するところの暗号鍵、「ワーク鍵」と成すことが可能になったのである。

【0026】このようなスキームをネットワークの中に

組み込むと、ユーザの扱い易い、暗号学上安全なストリーム暗号システムが可能となる。請求項3は、これを実現する乱数列の制御手段を述べたものである。すなわち、図13に示すように、例えば一方の端末側が請求項1又は2によって得た秘密鍵を用いてカオス的軌道を得た後に、このカオス軌道に基づいた乱数で平文をXORしてストリーム暗号にして送信し、他方の端末側では同じ秘密鍵を用いて同じカオス的軌道を発生させ、この軌道を用いて発生させた乱数と該ストリーム暗号とをXORしているので元の平文を得ることができるようになる。

【0027】また、請求項4は、One－Time Padを実現する目的で、暗号通信方式で「共有されている秘密鍵」すなわち共通鍵［X_1］を、当該通信の作法とタイミングを規定した暗号通信プロトコルの中に組み、その「生成」、「利用」、「更新」の時を持たせることを要旨としている。

【0028】請求項5、6、7は、秘密鍵を共通鍵にしてそれを「一回毎に捨てる」という具体的なメカニズムに関する。すなわち、請求項5は、請求項4の暗号通信方式にて、共通鍵［X_1］が当該暗号通信プロトコルによって「更新」される場合、当該暗号通信プロトコルは、共通鍵［X_1］のカオス的軌道［X_1、X_2 …X_i］に含まれる軌道値［X_{i+1}］を、新しい秘密鍵として指定し、該秘密鍵をもって共通鍵の更新を行なうことを要旨とする。

【0029】請求項6は、暗号通信方式にて、共通鍵の更新を、少なくとも二つの端末が同期して実施するために、該通信プロトコルは、通信状態のアプリケーションソフトを「送信モード」か「受信モード」のどちらかの状態に置く一方、そこで送受信される情報は、固定長部と可変長部とから成る統一された通信文フォーマットに載せられねばならず、その固定長部には「今回使用する共通鍵［X_1］」と「次回使用予定の共通鍵［X_{i+1}］」と「付加情報」の三つの情報ユニットが載り、対応した可変長部には「本文」が載ることを要旨とする。

【0030】請求項7は、暗号通信方式にて、通信プロトコルの送信モードでは、固定長部の「今回使用する共通鍵［X_1］」を用いて、共通鍵自体［X_1］と次回使用予定の共通鍵［X_{i+1}］及び付加情報の三つの情報ユニットを一緒に暗号化することを要旨とする。

【0031】以上5、6、7のメカニズムによって共通鍵が固定された不要の鍵ではなく、通信の毎に変化する「無常鍵」あるいはCurrent Keyとすることができるようになる。

【0032】請求項8は、セッション管理がクライアント、サーバー相方で実施されるメカニズムに関する。すなわち、暗号通信方式にて、受信モードの下で暗号文を受け取った端末は、自端末で保管していた共通鍵［X_1］.decによって（受信モード側の鍵を［X］.decと記号する）暗号文を復号し、復号された方の共通鍵のデータ［X_1］.encと（送信モード下の鍵を［X］.encと記号する）受信モード下の共通鍵［X_1］.decとを比較し、両者が等しければ該受信モードの端末は送信モードの端末を正当な暗号通信の契約者と認定する一方、両者が等しくなければ、受信モードの端末は送信者を「成りすまし」、「詐称」の可能性があるものとして例外処理を行なうか又は、送信者が「不正な通信歴」を経過した可能性があるとして例外処理を行なうなどして、共通鍵を通信プロトコルに組み入れたことによって送信者の「認証」を可能としたこと要旨とする。

【0033】これにより、もし他人の共通鍵を盗んで自分の端末からコネクションを張ってきた場合とか、他人のIPアドレスを使ってコネクションしてきた場合など、上の条件は成立しない。ここで、鍵ファイルを備えることが必須要件である。

【0034】請求項9は、6、7、8の請求項に該当する暗号通信方式にて、通信文を暗号化ファイルに保管する場合、固定長部の三つの情報ユニットを共通鍵で暗号化するのではなく、固定長部の付加情報部に本文を暗号化する際に使った秘密鍵を置き、固定長部全体の暗号化をICカード等に保管されている別途秘密鍵（マスター鍵）によって実施することを要旨とする。これは、各端末独自のファイルの暗号化保管とその通信とを融合したメカニズムに関する考案である。

【0035】これにより、秘密鍵を他人が知ることは不可能であるから、オープンネットワーク上での詐称等の犯罪が完全に抑制される。

【0036】

【発明の実施の形態】入れ籠構造を成したレジスターをハードウェアとして実現するか、ソフトウェアで実現するかによって実施の形態は大きく異なる。以下にソフトウェアによる実施の形態を記載する。

【0037】パソコンの乱数を作る有限精度空間は、そのレジスターの能力、すなわち2^{31}オーダーの2進数を記憶する能力で限定されていることを述べた（手段の項にて）。これを使って「可算個の無限」をどうやって表現するかを述べる。

【0038】このレジスターは少なくも「10^9 個」の整数を表現する能力をもっている。そこで、先に述べたように、可算個の無限（10^L , L→∞）を「10^9 個」のグループに分けて、可算個の無限の中に「10^9 を法とした入れ籠の構造」を導入する。

【0039】これをソフトウェアで表現すると、数値「0～10^9 －1」を一つの整数型の配列要素の中に収容し、その配列要素が多数整列した配列を定義する。そして、この配列を“親”として、その配列要素の一つを“子”と考えてこの“子”をさらに“親”と同様の配列となるように新たな配列を定義する。

【００４０】このようにして順次「入れ籠」を構成するのである。このようなアルゴリズムの実現はコンパイラー言語の得意とするところである。

【００４１】この構成の中で「10^9 を法とする」代りに「10^1 を法とした入れ籠の構造」を動入しても理論に変りはないことに着目する。

【００４２】この特殊なケース（ｍｏｄ）を我々は、たまたま１０進数と呼んでいるので、表面的には「１０進の数列」も「10^1 を法として、入れ籠の構造」も同じに見える。これは高速演算を実現する上で重要な着目点である。

【００４３】つまり、入れ籠の個数ｎを増やすことは、１０進数の桁数ｎを増やすことと等価になる。

【００４４】従って、通常の整数形の配列を大きくとることが、入れ籠の数を増やすことと同じ効果を生むので、すでに提出済みの出願「特願平８－１０８０５８」＊（周期性を考慮したカオス的乱数列の発生装置）では、実施の形態としてこの特殊な法（10^1 ）を採用した。

【００４５】アルゴリズムを詳細は、上記に譲って、そのアルゴリズムをシステム図としたのが図１であり、請求項１に対応するものである。これを見ると、初期値Ｘｉを「秘密鍵」として入力して（Ｓ１）、カオス的写像演算を行いながらカオス的軌道（$X_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n$ ）を次々と生成する（Ｓ２、Ｓ３）。そして、この生成が終了すると、その終了時点での演算値を次回の初期値として保存する（Ｓ４）。つまり、新しい秘密鍵として保存する。

【００４６】一方、ステップＳ２で生成された軌道を用いて一方向圧縮によるバイナリー乱数を生成し（Ｓ５）、これを乱数ストリームを暗号鍵（動的ワーク鍵）として出力する（Ｓ６）。

【００４７】すなわち、請求項１は、

（１）外部入力部；初期値設定→“秘密鍵”（Ｓ１）
　　カオス写像演算回数
（２）出力部；　乱数ビットストリーム　→“動的ワーク鍵”（Ｓ６）
　　最新の軌道値　→“新しい秘密鍵”（Ｓ４）
（３）処理部；　カオス的軌道生成部（Ｓ２）
　　カオス写像制御部（Ｓ３）
　　一方向圧縮関数部（Ｓ５）

とから構成されるシステムであることを主張している。

【００４８】通常、ワーク鍵は固定の長さ、５６ビットとか１２８ビットというように決められた長さで使われるが、このシステムでは、「カオス写像演算回数」を制御することによって、任意の長さのワーク鍵を生み出すことができる。従って、これを「動的ワーク鍵」と言う。

【００４９】これがストリーム暗号に最適な乱数生成システムであると主張するためには、さらに、この動的ワーク鍵が、他の秘密鍵によってコントロールできて、しかもどちらの鍵も同じパターンの鍵が実用上現われない、すなわち暗号学上安全であることと保証しなくてはならない。

【００５０】今、該レジスターの１０進収容能力を配列で１２８桁と設定してみよう。そうなると、カオス的軌道がループする平均のループ周期は「Ｃｏｍｐｕｔｅｒ ｓｍａｔｈ．ａｐｐｌｉｃ．ＶＯＬ２１ ＮＯ８ ｐｐ ９３－９４ １９９１ Ｇｒｅａｔ Ｂｒｉｔａｉｎ」（以下文献２という）に示すように、

【数１】

$$\text{周期} \propto 10^{\frac{128}{2}} = 10^{64} \approx 2^{192}$$

である；何を意味しているかと言うと初期値の個数は10^{128}個あり、同じ初期値が出現するのは、10^{64}回使った後である。

【００５１】これから作り出したワーク鍵も同じパター※ンに到達するには2^{192} ビット出しつくした後である。すなわち、アタックは極めて困難であることが判る。

【００５２】よって、ここに任意の初期値（上の例では、初期値の個数＝10^{128} 個）を「秘密鍵」として「動的なワーク鍵」を作り出す乱数生成装置及びシステムが得られたことになる。

【００５３】尚、このビットストリームは、０．０１％以下で「真正乱数」となることが確かめられた。真正乱数とは0と１とが等確率で出現する乱数である。実際にカオス写像がカオス的軌道を生成しているという一つの例証である。

【００５４】この乱数生成システムのもう一つユニークな長所は、乱数の「再現性」を保証していることである。再現性とは、コンピュータのハードウェアに依存しないこと、又ＯＳなどのプラットフォームに依存しないことで、しかも乱数を制御できることを言う。

【００５５】これが可能になった理由は、「１０進数を記憶するレジスター」をカオス写像の舞台に使っているからである。

【００５６】従来の乱数生成法では、２進のレジスター上にベルヌーイ・シフトやオーバーフローという手法を使っていたので、ハードウェアやソフトウェアに依存せざるを得なかったのである。

【００５７】さらにこの再現性をサポートしている理由がもう一つある。カオス写像は「それ自身への写像」であるから、もし10^9 を法としたら、全ての掛け算効果をこの中に戻さねばならない、つまり割り算を行なわね

ばならない。

【0058】例えば、1という数値と100という数値を10^9で割り算したら、コンピュータはその割り算結果を区別できないばかりか、ハード／ソフト毎に結果も異なってしまう。これではストレンジ・アトラクターはおろか、カオス的軌道にもならない。つまり、単に数学的定理をコンピュータでシミュレーションしても乱数にならないし、ましてや鍵は作れないということを意味する。

【0059】もし強引にそれらしくやっても、「同じ鍵が多数出現してしまう」のである。従来は、このような事態を避けるために、2進レジスターを援用してベルヌーイ・シフトとかオーバーフロー演算を使って、割り算を避けていたのである。もちろん、この場合は再現性が犠牲となる。

【0060】すなわち、請求項1は、これらのアポリアを一挙に解決した発明に関するものであると言える。

【0061】一方、請求項2は、実施の形態をシステム論から、請求した内容である。また本発明の乱数生成システムをネットワークに組み込み、暗号通信のプロトコルを考案したのが、請求項3～8である。

【0062】従来技術の項で解説したように乱数を一回使用する毎に捨てるという（One－Time Pad）を通信規約にしようとすると、該乱数生成システムの秘密鍵を「共通鍵」にして、「1回毎に捨てる」ということになる。では、鍵を捨ててしまった後、どうやって暗号通信をするかというところがこの考案のポイントの一つである。

【0063】請求項9は、イントラネットの欠陥を解決した発明である。企業の秘情報を外部の者に盗られないようにする技術は色々あるが、内部から情報発信してしまうとしたら、全く防ぎようがない。

【0064】イントラネットの使い易さを制限すれば解決つくかも知れないが、今度はオープンネットワークの利便性を犠牲にしてしまうことになる。そこで秘情報を常時暗号化したファイルの状態にして利用するというのがそのポイントであり、 又必要に応じて他の端末に共通鍵で配達するという自由自在性がもう一つのポイントである。

【0065】それでは、請求項3～9までの全体のスキームを図2，図3に示すと同時に、その通信規約（プロトコル）の概要を、図4、図5、図6、図7、図8に示して、実施の形態を述べる。

【0066】クライアント／サーバーモデル（以下C／Sモデルと略記する）を実施の形態にして、サーバー側を図2に、クライアント側を図3に表わす。基本的な仕組みはどちらも同じで、鍵の保管の仕方だけ異なる。

【0067】図2のサーバ側及び図3のクライアント側は、GUI10（Graphical User interface）と、鍵処理部11と、平文格納用のHDD12と、鍵ファイル格納用のHDD13と、乱数生成部14と、乱数である動的ワーク鍵格納用のHDD15と、暗号文格納用のHDD16と、暗号・復号部（XOR）17と、TC／IP部19とを備えている。

【0068】図3に示すクライアント側は「鍵」をICカード20に、図2に示すサーバー側はHDD13に鍵ファイルとして保管している。

【0069】暗号・復号部（XOR）17は、排他的論理和である。任意のファイルを乱数で“XOR”を作り、再び乱数で“XOR”すると元のファイルに戻る；これがストリーム暗号（対称暗号）の基本である。

【0070】図3に示すクライアント側の「平文のHDD12」にある平文ファイルをクライアント側のGUI部10で指定すると、その平文ファイルのビット数と等量の乱数ストリームがクライアント側の「乱数のHDD15」から取り出される。

【0071】これで暗号のファイルがもう一つの「暗号のHDD16」上に作られることとなる。

【0072】つまり、図2に示すサーバー側の「暗号のHDD16」の暗号ファイルは、

暗号ファイル←平文ファイル“XOR”乱数ストリーム

となる。

【0073】この暗号ファイルがクライアント側のTCP／IP部19により、図2に示すサーバー側の「暗号のHDD16」に配送されると、そしてもし同じ乱数ストリームがサーバー側にあるかじめ備わっていたとすると、

平文ファイル←暗号ファイル“XOR”乱数ストリーム

として、元の平文が現われる。

【0074】故に、同じ乱数ストリームを両端末で備える代りに、請求項1又は2の「乱数生成システムを両端末で備える一方、秘密鍵を保管したICカード20等の記憶媒体と、それを脱着自在とする入力装置を備え、あるいは端末の記憶装置に秘密鍵と、そのユーザーIDのテーブルからなる鍵ファイルを備え」、その秘密鍵も両端末で「共有する」としたのが請求項3である。

【0075】ここで重要な、見落してはならないポイントは、秘密鍵は両端末だけで共有されねばならないということである。当然のことのようであるが、乱数生成システムが従来のように短い周期でループするようなシステムであるとすると、同じ鍵が多数出現してしまうのである。従って、従来の乱数生成技術を同じ環境に組み入れても用を作さない。両端末だけで共有される鍵を「共通鍵」と言う。

【0076】従来の共通鍵方式の暗号通信は、鍵を割り出すのは時間の問題であるという欠点がある。その理由は、

① 毎回同じワーク鍵を使って暗号文を作っている。

② 短いワーク鍵で長い暗号文を作る場合がある。

【0077】一方、One－Time Pad暗号は、

同じ乱数ストリームを二度使わない、すなわち、同じワーク鍵を使わないということで、もっとも安全な暗号と考えられている。あるいは、安全であることを証明できる唯一の暗号であるといわれている。これを請求項３に適用すると、「ワーク鍵を使い捨てにすると同時に、ワーク鍵を作り出す共通鍵も使い捨てにする」ということになる。このようなことを可能とする唯一の方法は、請求項３の乱数生成システムが作り出す秘密鍵を「通信プロトコルの中に組み入れる」ことである。このときの秘密鍵が作り出したワーク鍵を「動的ワーク鍵」と言う。動的とは、ワーク鍵の長さが平文や暗号文に依存することと、”Ｏｎｅ－Ｔｉｍｅ”（１回しか使われない）であるという二つの意味である。

【００７８】つまり、要約して関数式にすると、

暗号文＝ｅ（平文、動的ワーク鍵）……（ａ）

平文＝ｄ（暗号文、動的ワーク鍵）……（ｂ）

動的ワーク鍵＝ｆ（共通鍵、平文、暗号文）……（ｃ）

として示される。

【００７９】さらに上述を土台にして請求項４～９が成立する。これを図式にすると図４、図５、図６、図７に表わすとおりである。これを用いて説明する。

【００８０】図４にて、クライアント（Ａ端末）がサーバー（Ｂ端末）に対してコネクションを張る送信モードの状態で、クライアント・ソフトは通信文全体のビット数を計算する（ｄ１）。通信文のフォーマットは図６に示すとおりである。この項目は請求項６に定義してある。通信文全体のビット数を今、ｎビットとすると（ＴＣＰ／ＩＰ部を除く）、これからカオス写像演算回数“ｉ”が決定される（ｄ２）。

【００８１】すなわち、１回の演算で一つの軌道値が作られ、その一つのカオス的軌道値から乱数を作るのであるが、何ビットの乱数を作るかは、一方向圧縮変換のアルゴリズムに依存する。

【００８２】現在使われているアルゴリズムＳＨＡ（Ｓｅｃｕｒｅ　Ｈａｓｈ　Ａｌｇｏｒｉｔｈｍ）では、出力は固定の１２８ビットである。

【００８３】一方、カオスビットは一つのカオス的軌道値から１ビット作られるのが原則であるので、通信文がｎビットならとなる。つまり、請求項１と２のＳＨＡの出力は変動する。すなわち”動的ワーク鍵”と言われる由縁である。

【００８４】一方、請求項１及び２のカオス的軌道の圧縮アルゴリズムをｇとすると、

ｉ＝ｇ（ｎ）

で与えられ、カオス写像演算回数“ｉ”が決まる。このｇの内容については請求外なので省略する。　＊

秘密鍵＝ｐ（前回の鍵、前回平文、前回暗号文）……（ｄ）

これで復号化する準備ができた（ｄ１７）。

【００９３】該動的ワーク鍵で暗号封筒を“ＸＯＲ”すれば元の平文に戻るはずである。しかし、元の平文に戻※るのは、次の必要条件が満たされている時だけである。

＊【００８５】共通鍵［X_1］.enc（送信側には・ｅｎｃを、受信側には・ｄｅｃの拡張子を付す）と（ｄ３）、上記“ｉ”からカオス的軌道

［X_1、X_2、X_3 …X_i］

が生成される（ｄ４）。これから通信文（図６）を暗号化するのに必要な動的ワーク鍵を作る（ｄ５）。

【００８６】そして、ｄ４で生成された乱数より次の軌道値を生成する（ｄ６）。次回の共通鍵となる秘密鍵は Xi のカオス写像から生成される（ｄ７）；

X_{i+1} ＝ｃ（X_i），ｃ；カオス写像

かくして

［X_i］.enc

［X_{i+1}］.enc

付加情報（今回無し）

本文

からなる平文ファイルを［X_i］.encから生成された動的ワーク鍵で１ビットずつ“ＸＯＲ”する（ｄ８）。

【００８７】これは丁度平文ファイルを“暗号の封筒”に入れたというように解釈してよい。この封筒を開封できるのは共通鍵［X_1］.decを持っている人だけである。

【００８８】上記“暗号封筒”をＴＣＰへ送り出した後（ｄ９）、クライアントは受信モードに変わる。ネットワーク端末Ａは図６に示すように該封筒をＴＣＰパケットにし、さらにＩＰパケットにして配送する。

【００８９】一方、端末Ｂ（サーバー）は、ＩＰパケットを開封し、次にＴＣＰパケットを開封して、“暗号封筒”を、受信モードにあるサーバーソフトへ渡す。

【００９０】図５にてサーバーソフトは、暗号封筒全体のビット数を数える一方（ｄ１０）、ＩＰパケットの“差出人”ＩＰアドレスを鍵ファイルへ送って、ＩＰアドレスに該当した共通鍵［X_1］.decを取り出す（ｄ１２）。

【００９１】また、暗号封筒全体のビット数はｎビットであるから

ｉ＝ｇ（ｎ）

として“ｉ”が求まり（ｄ１１）、これからサーバーソフトは、カオス的軌道［X_1、X_2 X_3 …X_i］を演算してカオス的乱数を生成させ（ｄ１３）、動的ワーク鍵を作る（ｄ１４）。そして、ｄ１３で生成された乱数に基づいて次回の軌道値を生成し（Ｓ１５）、

次回の秘密鍵［X_{i+1}］

X_{i+1} ＝ｃ（X_i）

より求める（ｄ１６）。

【００９２】結局、秘密鍵（カレント）は、次のような関数式で与えられる；

【００９４】［X_1］.enc（エンクリプション）＝［X_1］.dec（デクリプション）

すなわち、クライアントがもし他人の共通鍵を盗んで自分の端末からコネクションを張ってきた場合とか、他人のＩＰアドレスを使ってコネクションしてきた場合など、上の条件は成立しない。そこで、図5のセッション管理部で、鍵ファイルから取り出した［X_1］.decと、復号化した後のデータ［X_1］.encとを比較する（ｄ１９）。一致しない場合は、ただちにコネクションを切断する（ｄ20）。成りすまし、詐称というオープンネットワーク上の詐欺行為はこれで防げる。

【００９５】また、セッションが一致した場合は、次へ進み「次回の秘密鍵」［X_{i+1}］.decと［X_{i+1}］.encとを比較する（ｄ２１）。この比較は原則必要ではないが、共通鍵方式の暗号通信の信頼性を確立するために行＊なう。必要ではないという理由は、元の平文のビット数と暗号文（サーバーへ届いた）のビット数はＴＣＰ／ＩＰでは等しいことが保証されているからである。しかし、ハッカーによって偽造パケットの挿入及びパケットの変更、あるいはリプレイ攻撃といった介入によって

$$i = g(n)$$

の計算結果が異なってしまう場合も予想される。

【００９６】従って、「次回の秘密鍵」が一致しない場合は、サーバーはクライアントに再送を要求するなどの（ｄ２２）、例外処理を行なう。一致した場合は、共通鍵の更新を行なうと同時に、サーバーは送信モードに変わる。結局、新しい共通鍵は、次のような関数式で与えられる；

$$\text{共通鍵} = s(\text{次回秘密鍵}.\ enc = \text{次回秘密鍵}.\ dec) \cdots\cdots (e)$$

このようにして共通鍵で次々と使い捨てにされる（更新）。これは、ｕｎｉｘの標準技術になっているＯｎｅ－Ｔｉｍｅ　Ｐａｓｓｗｏｒｄ（ＯＴＰ）の効果を網羅する。

【００９７】クライアントとサーバーのやりとりを、もう少しマクロで見たのが図8である。図中の（２）ホームページをダウンロードすると、A端末の画面には、（３）本文入力や通信文作成の支援画面が提供される。（４）は図4で説明した内容である。サーバーは図5で説明したように、（５）認証、（６）復号化、（７）共通鍵の更新ということで鍵処理部１１が鍵ファイルを［X_1］.dec→［X_{i+1}］.decへ更新する。

【００９８】そして、送信モードへ入って、サーバーはクライアントへ返答を出す（８）。このときの通信文フォーマットも図6の通りである。この通信文フォーマットは、固定長部と可変長部とから成る統一された通信文フォーマットに載せられており、その固定長部には「今回使用する共通鍵［X_1］」と「次回使用予定の共通鍵［X_{i+1}］」と「付加情報」の三つの情報ユニットが載せられている。また、対応した可変長部には「本文」が載っている。

【００９９】従ってこの“暗号封筒”を受け取ったクライアントをサーバーと同じく、（９）認証、（１０）復号化、（１１）共通鍵の更新ということで、ＩＣカード２０の内容を［X_{i+1}］.enc→［x_{i+j}］.encへ更新する。以下同様の手続をくり返して終了に至る。クライアントは終了メッセージをやはり図6の通信文としてサーバーへ送る。

【０１００】この一連のＣ／Ｓモデルの通信で、共通鍵が図１１に示すように次々と変化して、使い捨てにされていることに注目する。

【０１０１】以上の説明から特徴的なスキームを拾い出す；

（１）乱数生成システムが通信プロトコルの中に組み込まれている。従って、その秘密鍵もワーク鍵もプロトコルの指示に従って「利用され」「生成され」そして「更※50 ※新される」。

（２）共通鍵は次々と「使い捨てにされる」。これは、色々な犯罪を抑止する。請求項３、４はこの（１）と（２）を請求しているものである。この内容を実現する具体論が請求項５、６及び７である。

（３）共通鍵を通信プロトコルに組み入れたことによって、セッション管理をクライアント、サーバーとも行なえるようになった。これにより成りすまし、詐称など従来の認証システムへの攻撃を抑止できる。

【０１０２】請求項８は、この（３）を請求したものである。通信文フォーマット（図６）が、固定長部と可変長部から構成されている理由を未だ述べていない。

【０１０３】インターネットツール類を特定の組織内で利用する場合、特に“イントラネット”と言われているが、社外の者が容易に社内へ侵入できてしまうので、色々なアクセス制御が工夫されている。しかし、いかにアクセス制御を工夫しても、社内の者が重要な機密情報を社外へ発信してしまうのを防ぐことはできない。そこで、重要なファイルはあらかじめ暗号化して保管する方法が残されている。例えば、鍵ファイルなどは暗号化して暗号ファイルにすべきものであろう。

【０１０４】例えば、既に特定の秘密鍵で暗号化されているファイルを指定して図４、図５、図６のスキームに載せるとしたら、暗号ファイルの秘密鍵を相手に伝える手段が必要となる。

【０１０５】そこで、図７に示すように、固定長部の付加情報部にその暗号ファイルの秘密鍵を載せることにすると、同一のスキームで任意の暗号ファイルを配送することが可能となることが判る。そうすると、図４に戻って、“暗号封筒”を作る際に

① 秘密鍵の入っている固定長部を暗号化すると同時に、暗号ファイルも、さらに共通鍵で暗号化する（図４のスキームではそうなる）。又は

② 固定長部のみ共通鍵で暗号化して、任意の暗号ファイルはそのまま固定長部にリンクする。

の二つのやり方が可能である。②の場合、これを受け取

った、図5のサーバーであれば、まず固定長部の復号化を行なわねばならない。

【0106】従って鍵を収納する固定長部は、あらかじめそのビット数が決められている必要があるのである。しかし、①の場合を採用すれば、必ずしも固定長である必要はない。どちらも、①又は②も可能なように、"暗号封筒"は固定長と可変長とから構成することとした。

【0107】

【発明の効果】以上のように本発明は、まず「入れ籠構造」を成した特殊レジスターを用いてカオス写像演算を実現し、この結果生み出されるカオス的軌道の任意の軌道値を秘密鍵とし、ワーク鍵として暗号通信に使用できるようにした。

【0108】また、このときの通信プロトコルは、この秘密鍵を共通鍵とする一方、該共通鍵を更新できることにしたので、オープンなネットワーク環境下にあるパソコン上で「One－Time　Pad暗号」を成立させること（共通鍵を使い捨てにする）ができるようになった。これで「盗聴」をパーフェクトに排除する効果が得られている。

【0109】さらに、送受信する情報の内、少なくとも固定長部には今回使用する共通鍵と次回使用予定の共通鍵という鍵情報が入っているので二重のセッション管理が行われる。その結果、「始点アドレスの偽造」、「パケットの変更、挿入」、「成りすまし」、「詐称」というオープンネットワーク上の詐欺行為がこれで排除されるという効果が得られている。

【図面の簡単な説明】

【図1】本発明の実施の形態の概略構成図である。

【図2】本発明を適用したサーバ側の概略構成図である。

【図3】本発明を適用したクライアント側の概略構成図である。

【図4】送信モード時の動作を説明する説明図である。

【図5】受信モード時の動作を説明する説明図である。

【図6】暗号封筒を説明する説明図である。

【図7】暗号封筒を説明する説明図である。

【図8】本発明による共通鍵の使い捨て通信プロトコルを説明する説明図である。

【図9】鍵階層を説明する説明図である。

【図10】ストリーム暗号の概念を説明する説明図である。

【図11】共通鍵が使い捨てにされることを説明する説明図である。

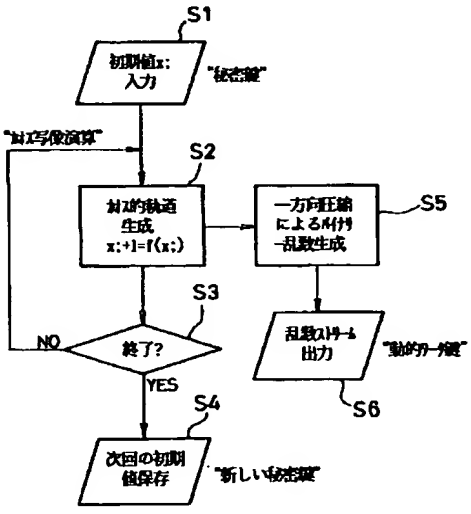
【図12】手段を説明する説明図である。

【図13】手段を説明する説明図である。

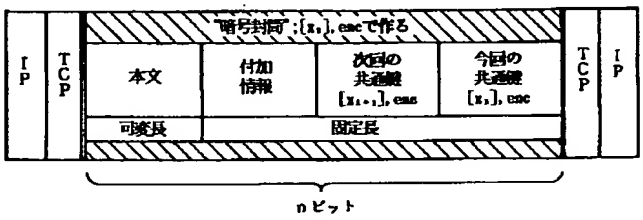
【符号の説明】

10　GUI
11　鍵処理部
12　平文格納用のHDD
13　鍵ファイル格納用のHDD
14　乱数生成部
15　動的ワーク鍵格納用のHDD
16　暗号文格納用のHDD
17　暗号・復号部（XOR）
19　TC／IP部

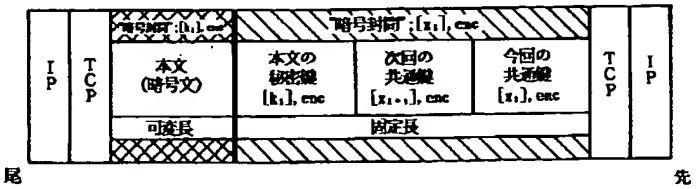
【図1】



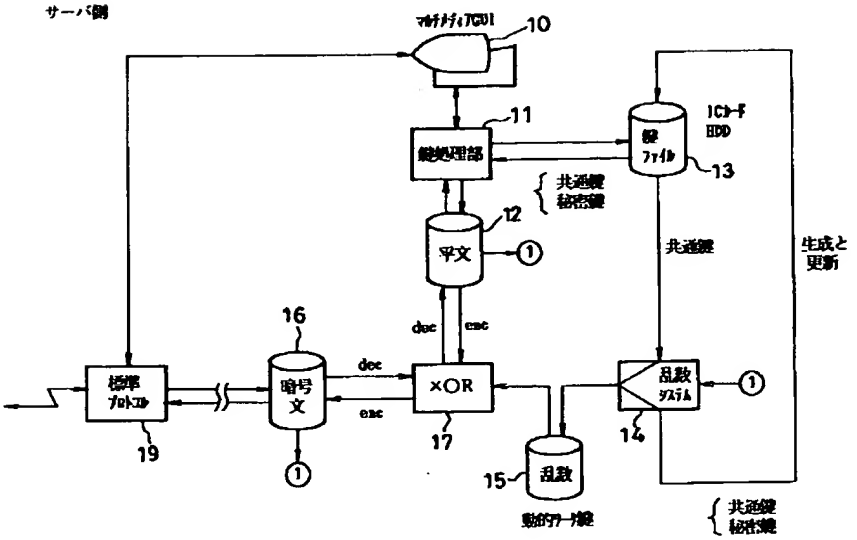
【図6】



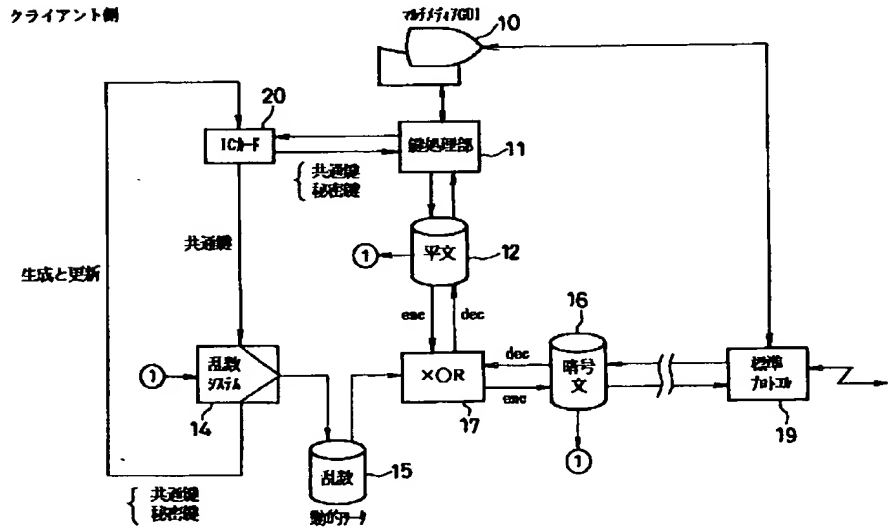
【図7】



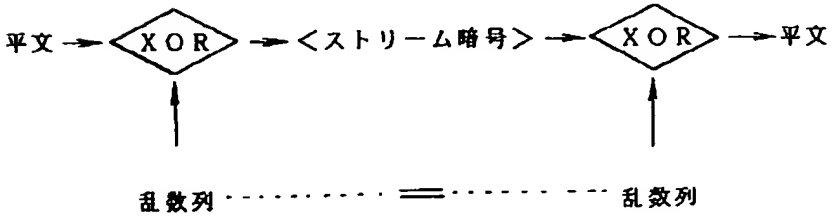
【図2】


サーバ側
10
11
12
13
16
17
14
15
19
平文
暗号文
XOR
乱数
共通鍵
秘密鍵
生成と更新
dec
enc

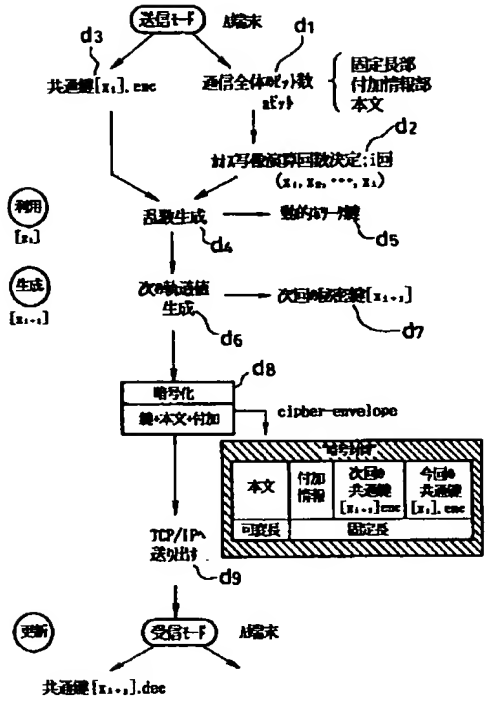
【図3】


クライアント側
10
20
11
12
13
14
15
16
17
19
平文
暗号文
XOR
乱数
共通鍵
秘密鍵
生成と更新
dec
enc

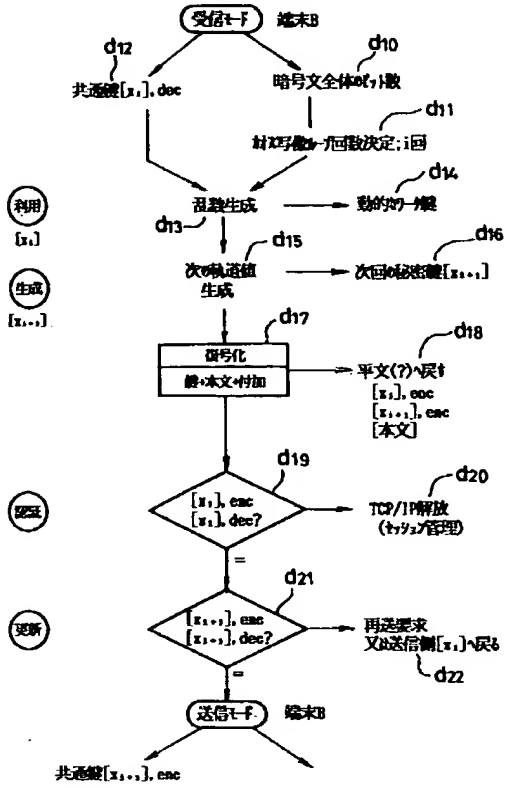
【図10】


平文
XOR
<ストリーム暗号>
XOR
平文
乱数列
乱数列
equal

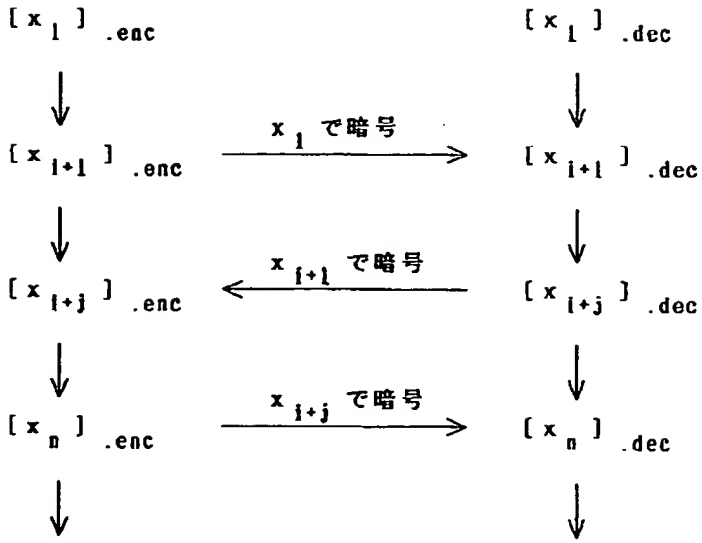
【図4】


送信ﾄﾞﾗ
d3
d1
共通鍵[x1], enc
通信全体のﾃﾞｰﾀ数
固定長部
付加情報部
本文
d2
d4
d5
d6
d7
d8
暗号化
鍵+本文+付加
cipher-envelope
本文
付加情報
次回の共通鍵[x1+1]enc
今回の共通鍵[x1], enc
可変長
固定長
d9
TCP/IPへ送り出す
受信ﾄﾞﾗ
共通鍵[x1+1], dec

【図5】

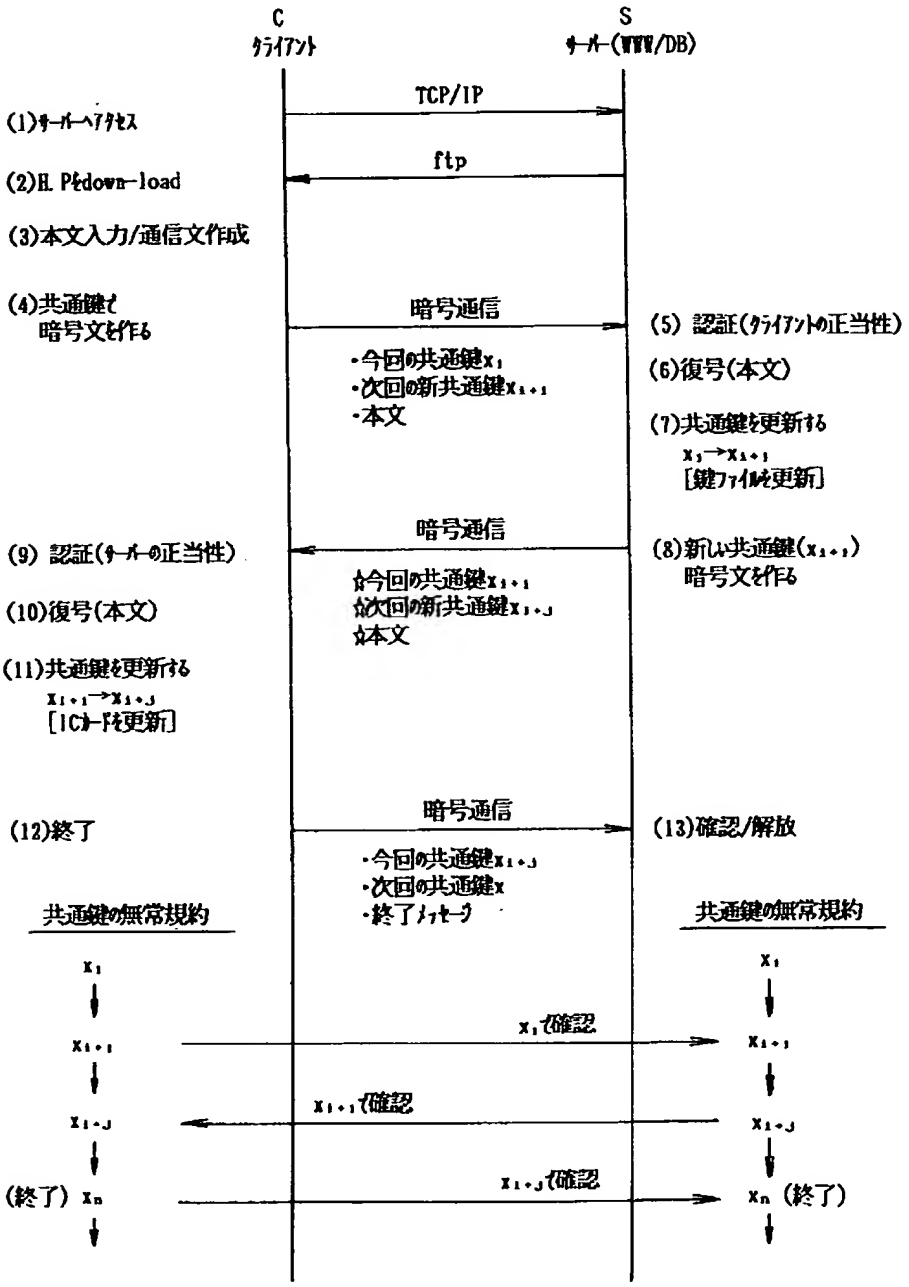

受信ﾄﾞﾗ
端末B
d12
d10
共通鍵[x1], dec
暗号文全体のﾃﾞｰﾀ数
d11
d14
d13
d15
d16
d17
d18
復号化
鍵+本文+付加
平文(?)へ戻す
[x1], enc
[x1+1], enc
[本文]
d19
d20
[x1], enc
[x1], dec?
TCP/IP解放
=
d21
[x1+1], enc
[x1+1], dec?
再送要求
又は送信側[x1]へ戻る
d22
送信ﾄﾞﾗ
端末B
共通鍵[x1+1], enc

【図11】


[x1].enc
[x1].dec
x1 で暗号
[x1+1].enc
[x1+1].dec
x1+1 で暗号
[x1+j].enc
[x1+j].dec
x1+j で暗号
[xn].enc
[xn].dec

【図8】

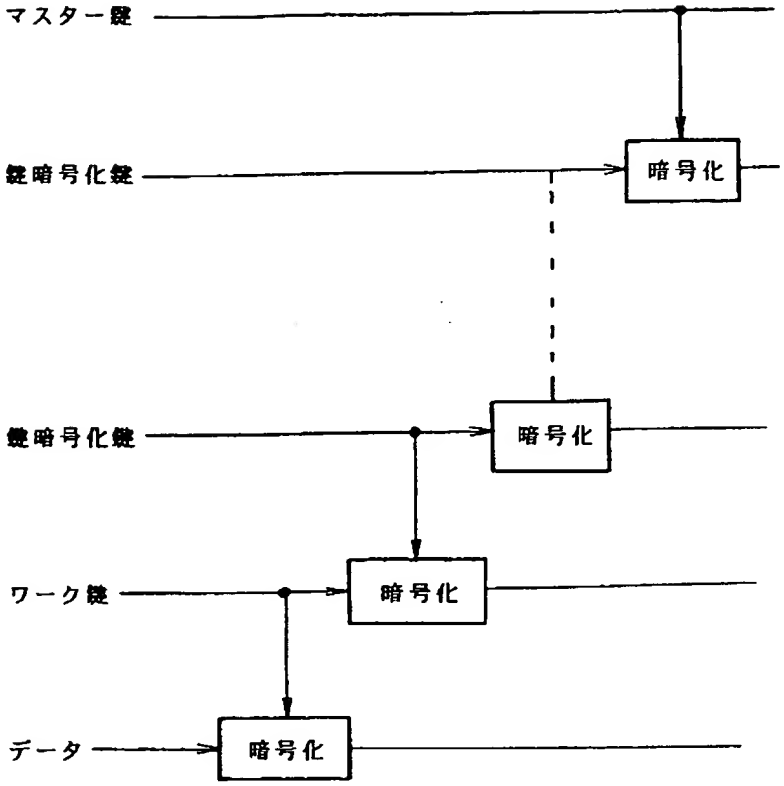
共通鍵の使い捨て通信プロトコル



【図12】

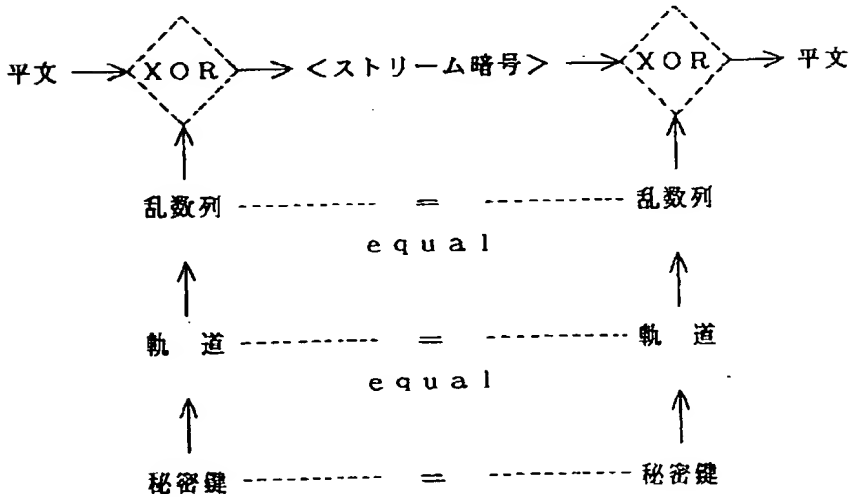
連続量 ― 可算個の無限 …①
　　└― 非可算個の無限 …②

【図9】


マスター鍵
鍵暗号化鍵
暗号化
鍵暗号化鍵
暗号化
ワーク鍵
暗号化
データ
暗号化

鍵階層

【図13】


平文
XOR
<ストリーム暗号>
XOR
平文
乱数列
＝
乱数列
equal
軌　道
＝
軌　道
equal
秘密鍵
＝
秘密鍵